Modal

Modal The Modal component is a powerful overlay system that extends the dialog service to provide custom modal dialogs with full content projection capabilities. It allows developers to create sophisticated modal interfaces with custom content, layouts, and interactions while maintaining consistent styling and behavior. The modal system is built on top of the AavaDialogService and provides a flexible foundation for creating various types of modal dialogs, from simple confirmations to complex forms and content displays. How to use ■ import { AavaDialogService } from "@aava/play-core" ; // Inject the service constructor (private dialogService : AavaDialogService) { } // Open a custom modal this . dialogService . openModal (MyCustomModalComponent , { width : '600px' , maxWidth : '90vw' }) ; Modal Variants ■ Simple Modal Features ■ Header Section : Custom header with icon and title Body Content : Descriptive text content Footer Actions : Primary and secondary action buttons Responsive Design : Adapts to different screen sizes Accessibility : Proper ARIA attributes and keyboard navigation Feedback Modal Features ■ Visual Feedback : Success icon with color-coded styling Input Fields : Text input for user feedback Centered Layout : Optimized for form interactions Action Buttons : Clear primary and secondary actions User Experience : Intuitive feedback collection flow Scrollable Modal Features ■ Content Scrolling : Handles overflow content gracefully Form Elements : Checkbox and form controls Structured Content : Organized sections with headings Flexible Height : Adapts to content length Footer Positioning : Actions remain accessible during scroll Features ■ Content Projection ■ The modal system uses Angular's content projection to provide flexible content structure: [dialog-header] : Header section for titles, icons, and navigation [dialog-body] : Main content area for forms, text, or components `[dialog-footer] : Footer section for action buttons and controls Flexible Sizing ■ Width Control : Customizable width with responsive constraints Height Management : Automatic height adjustment or fixed heights Responsive Behavior : Adapts to different screen sizes Max Dimensions : Configurable maximum width and height Service Integration ■ DialogService : Centralized modal management Component Injection : Dynamic component rendering Configuration Options : Flexible modal configuration Lifecycle Management : Proper cleanup and resource management Accessibility ■ Keyboard Navigation : Full keyboard support (Tab, Escape, Enter) Screen Reader : Proper ARIA attributes and labels Focus Management : Focus trapping and restoration High Contrast : Maintains accessibility standards API Reference ■ AavaDialogService.openModal() ■ openModal (component : Type < any > , config ? : ModalConfig , data ? : any) : Promise < any > ModalConfig Interface ■ Property Type Default Description width string - Modal width (e.g., '600px') maxWidth string - Maximum width constraint height string - Modal height maxHeight string - Maximum height constraint showCloseButton boolean true Show close button backdrop boolean true Show backdrop overlay closeOnBackdrop boolean true Close on backdrop click Content Projection Attributes ■ Attribute Description [dialog-header] Header section for modal title and navigation [dialog-body] Main content area for modal content `[dialog-footer] Footer section for action buttons Best Practices ■

Modal Design ■ Clear Purpose : Make modal purpose obvious through title and content Appropriate Sizing : Choose modal sizes that fit content without overwhelming Consistent Layout : Use consistent header, body, and footer structure Visual Hierarchy : Maintain clear visual hierarchy within the modal Responsive Design : Ensure modals work well on all screen sizes Content Organization ■ Header Content : Include clear titles and optional icons Body Structure : Organize content logically with proper spacing Footer Actions : Place primary actions on the right, secondary on the left Content Length : Keep content concise or implement scrolling for long content Form Elements : Use appropriate form controls and validation User Experience ■ Clear Actions : Make button actions and consequences clear Escape Options : Always provide a way to close the modal Loading States : Show appropriate loading indicators for async operations Error Handling : Provide clear error messages and recovery options Accessibility : Ensure keyboard navigation and screen reader support Performance ■ Lazy Loading : Load modal content only when needed Component Reuse : Reuse modal components when possible Memory Management : Properly clean up modal instances Change Detection : Use OnPush strategy for optimal performance Bundle Optimization : Import only needed modal components Styling ■ The modal component uses CSS custom properties for theming: :root { --modal-background : #ffffff ; --modal-border : #e2e8f0 ; --modal-shadow : 0 20 px 25 px -5 px rgba (0 , 0 , 0 , 0.1) ; --modal-radius : 8 px ; --modal-padding : 24 px ; --modal-header-color : #3b3f46 ; --modal-body-color : #6b7280 ; --modal-footer-border : #f1f5f9 ; }

■ No code found